View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Theseus

VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Ari-Matti Huhtamäki

WEB-SOVELLUS SAMANHENKISTEN KÄYTTÄJIEN LÖYTÄMISEKSI

Tekniikka ja liikenne
2013

VAASAN AMMATTIKORKEAKOULU

Tekniikka ja liikenne
2013

Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Ari-Matti Huhtamäki
Opinnäytetyön nimi	Web-sovellus samanhenkisten käyttäjien löytämiseksi
Vuosi	2013
Kieli	suomi
Sivumäärä	37
Ohjaaja	Pirjo Prosi

Opinnäytetyön aiheena on web-sovellus, jossa käyttäjä rekisteröityy ja aktivoi käyttäjätilinsä kirjautuakseen käyttämään sovellusta. Sovellus esittää käyttäjälle kysymyksiä, joihin vastaamalla sovellus vertaa muiden käyttäjien vastauksia vastattuihin kysymyksiin ja laskee niiden perusteella prosentuaalisesti vastausten yhtäläisyyksiä. Eniten kysymyksiin yhtenevästi vastanneisiin muihin käyttäjiin voi käyttäjä ottaa yhteyttä sähköpostitse.

Sovelluksessa käytettiin ohjelmointikielenä ja tekniikoina pääosin PHP:tä ja HTML:ää sekä myös CSS:ää ja jQueryä ja Ajaxia. Tietokantana toimi MySQL ja HTTP-palvelinohjelmana Apache2.

Koska oppilaitoksessa ei opetettu PHP:ta opettelin kielen omatoimisesti, joka hieman pitkitti työn suoritusta. Se ei kuitenkaan ollut este kaikkien toiminnallisuuksien tekemiseen. Myös vastausten vertaileminen oli todella haastava, mutta se onnistui lopulta hienosti. Työ kaiken kaikkiaan sujui mainiosti vaikka hieman ohjelmointi pitkittyi haasteiden tullessa vastaan.

Avainsanat HTML, PHP, kysymysten käsittely, MySQL, Ajax

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikan koulutusohjelma

ABSTRACT

Author	Ari-Matti Huhtamäki
Title	Web-Application for Finding Congenial Users
Year	2013
Language	Finnish
Pages	37
Name of Supervisor	Pirjo Prosi

This thesis presents a web application for finding congenial users via questions. The user must register and activate their useraccount for being able to login to the web-application. The application asks user questions and by answering to those questions the application calculates the persentage of matchups from other users similar answers to the exact questions. User gets list of top 5 users found by matchup and he/she can contact those users via email.

Programming languages and techniques used for building this web-application is mainly PHP and HTML. Also CSS, jQuery and Ajax were used. MySQL was used for database and Apache2 was used for HTTP-server.

Because our learning institution did not teach PHP specifically I had to learn PHP on my own. It was not an obstacle for finishing this thesis but it held slightly back at time schedule. Also matching up answers were a challenge and it took slightly more time than was allocated to be used for it. Overall the whole thesis process was good and it was interesting to complete it.

Keywords HTML, PHP, question management, MySQL, Ajax

SISÄLLYSLUETTELO

TERMISTÖ

Alla lueteltuna yleisimmät käytetyt lyhenteet ja niiden merkitykset.

Ajax	Asynchronous JavaScript And Xml. Joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia.
Apache	HTTP-palvelinohjelma staattisten tiedostojen jakamiseen HTTP-protokollan yli.
CSS	Cascading Stylesheets. Www-sivujen tyylien määrittelemiseen käytetty tyyliohjekieli, jolla useat tyylimääritykset voidaan yhdistää säännöstöksi.
HTML	Hypertext Markup Language. Merkintäkieli, jota käytetään web-sivustojen rakentamiseen.
JavaScript	Oliopohjainen komentosarjakieli, jota käytetään web-sivustojen dynaamisuuden luomiseen.
LAMP	Linux, Apache, MySQL, PHP. Avoimeen lähdekoodiin perustuva web-palvelimen muodostava ohjelmistokokonaisuus.
Linux	Unixin kaltainen käyttöjärjestelmä, joka alun perin syntyi Unixin kopioksi. Ei kuitenkaan pohjaudu Unixiin.
MySQL	SQL-standardiin perustuva, mutta laajennettu tietokannan hallintahärjestelmä. Yksi suosituimmista web-sovelluksissa käytetyistä tietokannoista.
PHP	Hypertext Preprocessor. Palvelinympäristössä käytettävä ohjelmointikieli, jota käytetään dynaamisten web-sivustojen toteuttamiseen.

SQL	Structured Query Language. Standardoitu relaatiotietokantojen hallintaan tarkoitettu kyselykieli, jota käytännössä kaikki relaatiotietokannat ymmärtävät.
Unix	Laitteistoriippumaton käyttöjärjestelmä, johon monet muut käyttöjärjestelmät pohjautuvat. Käytetään yleensä palvelimissa.
W3C	World Wide Web Consortium. Www-standardeja ja suosituksia kehittävä kansainvälisten yhteisöjen ja yritysten yhteenliittymä.

KUVA- JA TAULUKKOLUETTELO

1 JOHDANTO

Työn aiheena on web-sovellus, jossa samanhenkiset käyttäjät voivat anonyymisti löytää ja ottaa yhteyttä toisiinsa keskustellakseen heille kiinnostavista ja tärkeistä asioista. Vaikka saman tyylisiä sovelluksia on jo olemassa, ei kuitenkaan tällaiseen tarkoitukseen olevaa sovellusta ole vielä saatavilla. Esimerkkejä saman tyylisistä sovelluksista ovat mm. deittisivustot, joiden ensisijainen tarkoitus on löytää käyttäjille elämänkumppani. Tämä sovelluksen tarkoituksena ei kuitenkaan ole löytää käyttäjälle kumppania, vaan tarkoituksena on löytää käyttäjälle samoista asioista kiinnostunut henkilö ikään, sukupuoleen tai kansallisuuteen katsomatta. Käyttäjät voivat keskustella vaikka vain hetken aikaa molemmille mieluisista aiheista tai vaihtaa sähköpostiosoitteita pidempiaikaista yhteydenpitoa varten.

Käyttääkseen sovellusta käyttäjän täytyy rekisteröityä palveluun syöttäen haluamansa käyttäjänimen, salasanan ja sähköpostiosoitteen. Rekisteröidyttyä käyttäjälle lähetetään sähköpostiin aktivointiosoite, jota klikkaamalla hänelle avautuu sivu, jolla käyttäjätilin voi aktivoida. Aktivoinnin jälkeen käyttäjä voi kirjautua sisään palveluun. Kirjauduttuaan palveluun käyttäjältä kysytään 20 pakollista kysymystä, joihin vastaamalla ja joiden perusteella käyttäjälle esitetään lisää kysymyksiä. Vastatessa kysymyksiin käyttäjälle näytetään lista viidestä eniten samankaltaisesti vastanneista käyttäjistä, joiden käyttäjänimeä painamalla käyttäjä voi lähettää sähköpostiviestin kyseiselle käyttäjälle.

2 KÄYTETYT TEKNIIKAT JA OHJELMISTOT

Tässä luvussa käydään läpi opinnäytetössä käytettyjä eri ohjelmointikieliä ja ohjelmointitekniikoita. Kappaleessa käydään läpi myös, mitä nämä ohjelmointikielet ja ohjelmointitekniikat ovat ja kerrotaan miten niitä käytetään opinnäytetyön tekemiseen.

2.1 HTML

Hypertext Markup Language, joka paremmin tunnetaan lyhenteellä HTML, on avoimesti standardoitu kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä eli hypertekstiä. Hyperteksti mahdollistaa hyperlinkeiksi kutsutut ristiviittaukset eri dokumenttien välillä. HTML:llä voidaan myös merkitä tekstin rakenne, esimerkiksi mikä osa tekstistä on otsikkoa ja mikä leipätekstiä. HTML tunnetaan erityisesti kielenä, jolla verkkosivuja luodaan.

Taulukosta 1 näkee HTML-kielen perusrakenteen, johon kuuluu <HTML>, <HEAD> ja <BODY>. Dokumentti määritellään HTML-tiedostoksi tagien <HTML></HTML> avulla. Dokumentissa tulisi olla myös otsikko-osa <HEAD></HEAD>, johon kirjoitetaan dokumentin otsikko tagien <TITLE></TITLE> väliin. Dokumentin varsinainen sisältö kirjoiteteaan tagien <BODY></BODY> väliin. HTML-sandardi edellyttää lisäksi dokumentin alkuun DOCTYPE-elementtiä, jolla ilmoitetaan dokumentissa käytetty HTML-standardi.

Taulukko 1. HTML-kielen perusrakenne.

```
<!DOCTYPE HTML PUBLIC"-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Otsikko</title>
</head>
<body>
Sivun sisältö tulee tähän.
</body>
</html>
```

HTML:stä löytyy paljon tietoa ja esimerkkejä, mutta jotain asioita on hyvä muistaa. Alla lista hyvistä HTML vinkeistä.

1. HTML:n tiedostopäätteitä ovat .htm ja .html

2. HTML-tageissa voi käyttää pieniä tai isoja kirjaimia (<html> tai <HTML>). On kuitenkin hyvä säilyttää sama tyyli kaikissa dokumenteissa käyttäen ainoastaan toista muotoa.

3. Kommentteja ja huomautuksia HTML:ssä voi lisätä <!-- ja --> tageilla, esim. <!-- Tämä on kommentti -->. Kommentit helpottavat dokumentin lukemista, mikä taas helpottaa dokumentin jatkokehitystä.

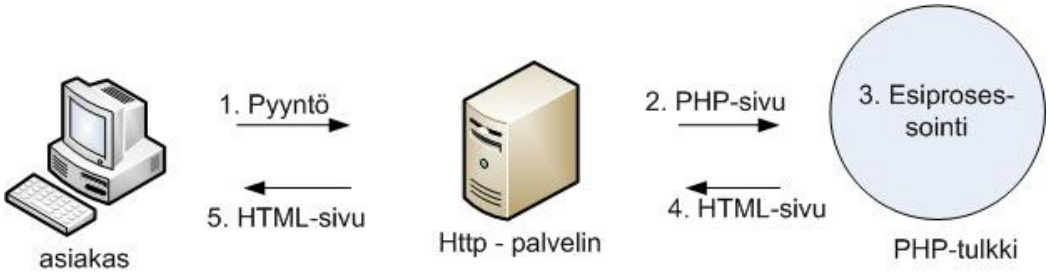
4. Annetut komennot täytyy muistaa myös sulkea konfliktien välttämiseksi (<html><body></body></html>).

Opinnäytetyössä käytettiin HTML:ää sivujen rakenteelliseen muotoiluun ja muiden tekniikoiden ja kielten avulla muodostui kokonaisuudesta toimiva sovellus.

2.2 PHP

PHP (Hypertext Preprocessor) on ohjelmointikieli, jota käytetään erityisesti web-palvelinympäristöissä dynaamisten web-sivustojen luomiseen. PHP on komentosarjakieli eli skriptikieli, joka tarkoittaa että ohjelmakoodi tulkitaan vasta ohjelman suoritusvaiheessa. PHP on tarkoitettu ainoastaan palvelinpuolen ohjelmointiin ja ohjelmakoodi tulkitaan vasta ohjelman suoritusvaiheessa palvelinpuolella. PHP on yksinkertainen, mutta erittäin monipuolinen ja tehokas ohjelmointikieli. Kieli on yleiskäyttöinen, mutta käytännössä se on erikoistunut web-ohjelmointiin. Kielen syntaksi muistuttaa C:tä ja Perliä. Ohjelmointikielen lisäksi PHP-ympäristössä on laaja luokkakirjasto. PHP:tä voidaan käyttää useilla eri alustoilla ja käyttöjärjestelmillä. Sen ensimmäinen versio julkaistiin vuonna 1994, ja nykyisin PHP on vertailussa johtava dynaamisten web-palveluiden tuottamiseen tarkoitettu ohjelmointikieli. Sen viimeisin vakaa julkaisu 5.4.5

julkaistiin 19.7.2012. PHP:tä käytetään HTML-kieleen sulautettuna, eli PHP:ta suorittaessa mukana tuleva HTML-osa jätetään käsittelemättä ja vain PHP-osa tulkataan ja suoritetaan. PHP-koodista saatava tulostus yhdistetään suorituksen jälkeen HTML-koodiin ja lähetetään käyttäjälle. (Zandstra M. 2005, 20)



Kuva 1. PHP-koodin toimintakaavio.

Kuvasta 1 näkee PHP-koodin toimintakaavion. Vaiheessa 1. asiakas esittää selaimella pyynnön PHP-sivusta HTTP-palvelimelle, palvelin tunnistaa tiedostopäätteestä millainen tiedosto on kyseessä. Vaiheessa 2. HTTP-palvelin irrottaa PHP osion HTML:stä ja siirtää sen PHP-tulkille. Vaiheissa 3. ja 4. PHP-tulkki suorittaa PHP-koodin ja muokkaa sen HTML-muotoon palauttaakseen sen HTTP-palvelimelle. Vaiheessa 5. HTTP-palvelin lisää PHP-tulkilta saamansa HTML-muotoisen koodin alkuperäiseen pyyntöön ja palauttaa sen asiakkaalle.

PHP:ta lisätään HTML:ään <?php ?>-tagien avulla. Aloittaakseen PHP-osan käytetään <?php tagia ja lopettaakseen PHP-osan käytetään ?>-tagia. Tageilla kerrotaan palvelimelle, että mitkä osat palvelin suorittaa ja mitkä osat selain suorittaa. Taulukosta 2 näkee miten PHP-osa upotetaan HTML:n sisään.

Taulukko 2. PHP:ta upotettuna HTML:n sisään.

```
<html>
<head>
<title>PHP-testi</title>
</head>
<body>
<?php echo '<p>Hei maailma!</p>'; ?>
</body>
</html>
```

Opinnäytetyössä lähes kaikki toiminallisuudet on toteutettu PHP:lla. PHP:lla on siis todella suuri osuus tässä työssä. Huomioitavaa on myös se, että en ole opiskellut lainkaan PHP:ta oppilaitoksessa, vaan olen opetellut PHP:n täysin omatoimisesti.

2.3 JavaScript

JavaScript on komentosarjakieli eli skriptikieli, joka on tehty helpottamaan dynaamisuuden kehittämistä web-sovelluksille. Kielellä voi luoda erilaisia toiminnallisuuksia ja tarkistuksia HTML-elementteihin, kuten tekstikenttiin ja nappeihin. JavaScript sisältää erilaisia tapahtumankäsittelyyn luotuja valmiita funktioita, joihin web-käyttöliittymän toiminta perustuu. JavaScriptiä ei tule sekoittaa Javaan.

Opinnäytetyössä JavaScriptiä käytettiin jQuery-kirjastojen ja Ajax-tekniikan käyttämiseen.

2.4 jQuery/Ajax

jQuery on kaikille selaimille tarkoitettu ilmainen, avoimen lähdekoodin JavaScript-kirjasto. jQueryn syntaksi on helppo ymmärtää ja se tekee siitä erittäin suositun. Parhaiten jQuery sopii toimintojen käsittelyyn, animaatioiden tekemiseen, DOM-elementtien valitsemiseen ja Ajax-sovelluksien toteuttamiseen. Se julkaistiin vuonna 2006 ja on nykyään maailman suosituin JavaScript-kirjasto. Se on erillinen JavaScript-tiedosto ja sisältää kaiken tarvittavan jQueryn käyttöönottoon.

Ajax eli Asynchronous JavaScript And XML on joukko web-sovelluskehityksen teknikoita, joiden avulla web-sovelluksista voi tehdä dynaamisia. Ajax ei tuo uutta, vaan se yhdistää jo olemassa olevia tekniikoita. Näitä tekniikoita ovat JavaScript, CSS, XML, XMLHttpRequest-objekti ja XHTML (tai HTML). Ajaxissa selainohjelma vaihtaa taustalla pieniä määriä dataa palvelimen kanssa niin, ettei koko verkkosivua tarvitse ladata uudelleen joka kerta käyttäjän tehdessä

muutoksen.

Opinnätetyössä jQueryä ja Ajaxia käytetään yhdessä kutsumaan taustalla PHP-koodia, lataamatta koko sivustoa uudelleen. Se siis mahdollistaa sen, että koko sivustoa ei tarvitse ladata uudelleen joitain toimintoja tehdessä.

2.5 SQL / MySQL

SQL eli Structured Query Language on kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä. SQLin avulla voidaan siis hakea, lisätä, poistaa ja muokata tietoa. Se on standardoitu, mutta siitä on olemassa erilaisia versioita. Peruskomennot, kuten SELECT, UPDATE, DELETE, INSERT, WHERE ovat tuettuja suurimmissa SQL-ohjelmistoissa. Taulukosta 3 näkee yksinkertaisen SQL-kyselyn, jota on käytetty opinnäytetyössä. (W3schools 1999-2013)

Taulukko 3. Esimerkki opinnäytetyössä käytetystä SQL-kyselystä.

```
SELECT *
FROM users
WHERE UNAME='$myusername'
AND PASSWD='$mypassword'
```

Opinnäytetyössä käytettiin MySQL-tietokantaa käyttäjätietojen, kysymysten, vastausten ja kysymysten kategorioiden tallentamiseen. Kuvasta 3 näkee opinnäytetyössä käytetyt taulut ja kyseisten taulujen sarakkeet.

3 SUUNNITTELU

Tässä luvussa käydään läpi ohjelman suunnitteluvaihe.

3.1 Vaatimusmäärittely

Vaatimusmäärittelyssä selvitetään, mitä ohjelmalla halutaan tehdä eli mitä ominaisuuksia ohjelman täytyy pitää sisällään ja mitä ominaisuuksia siltä vaaditaan. Alla lista tärkeimmistä ominaisuuksista, joita ohjelman täytyy pitää sisällän.

Sisäänkirjautuminen.

Rekisteröinti.

Kysymysten syöttäminen.

Kysymyksiin vastaaminen.

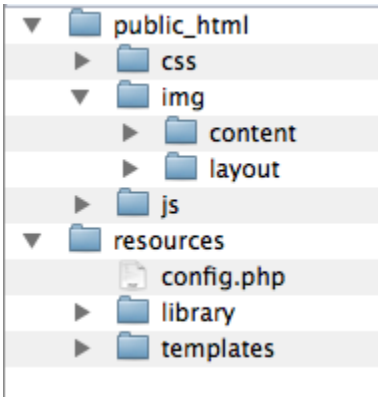
Vastausten vertailu ja tulosten esittäminen.

Yhteydenotto käyttäjien välillä.

3.2 Tiedostorakenne

Kun tärkeimmät ominaisuudet oli määritelty alkoi ohjelman tiedostorakenteen suunnitteleminen. Tiedostorakenteen suunnittelussa oli otettava tietoturva-asioita huomioon. Hain tietoa Internetistä PHP-projektien yleisistä tiedostorakenteista ja sieltä esiin tuli yksi erityisen hyväksi osoittautunut tiedostorakenne. Tässä tiedostorakennemallissa on kaksi erillistä kansiota. Toinen kansioista sisältää kaikki tiedostot, joihin käyttäjillä on pääsy selaimen kautta. Tämän kansion nimi on public_html ja se sisältää alikansioita css, js ja img. Img-kansio sisältää vielä kaksi alikansioita content ja layout. Toinen juurikansioista on taas kaikkia sellaisia tiedostoja varten, johon käyttäjällä ei ole suoraan pääsyä selaimella. Tämä kansio on nimeltään resources ja se sisältää tärkeitä ja arkaluontoisia tiedostoja, kuten

config.php, jossa määritellään mm. tietokantayhteyden asetukset. Resouces-kansio sisältää vielä kaksi alikansiota library ja templates. Kuvasta 2 nähdään visuaalinen puurakenne työn tiedostoista. (Nettuts+/Organize Your Next PHP Project the Right Way, 2013)



Kuva 2. Tiedostorakenne.

Public_html-kansio on määriteltynä Apache:n konfiguraatioissa siten, että kun sivuston osoite kirjoitetaan selaimen osoitekenttään se osoittaa palvelimella kansioon public_html (tiedostoon /public_html/index.php pääsee käsiksi osoitteesta esimerkkidomain.com/index.php). Alla listaus ja selvennys public_html-kansion sisältämistä kansioista. (Nettuts+/Organize Your Next PHP Project the Right Way, 2013)

css – Sisältää kaikki css-muotoilutiedostot.

img – Sisältää kaikki kuvatiedostot. Jaoteltu sisällön kuvatiedostoihin (content) ja ulkoasun kuvatiedostoihin (layout).

js – Sisältää kaikki JavaScript tiedostot.

Resources-kansio sisältää kaikki konfiguraatiotiedostot, kustomoidut luokkakirjastot ja muut tiedostot, jotka sovelluksessa toimivat resursseina. Alla listaus ja selvennys resources-kansion sisältämistä kansioista ja tiedostoista. (Nettuts+/Organize Your Next PHP Project the Right Way, 2013)

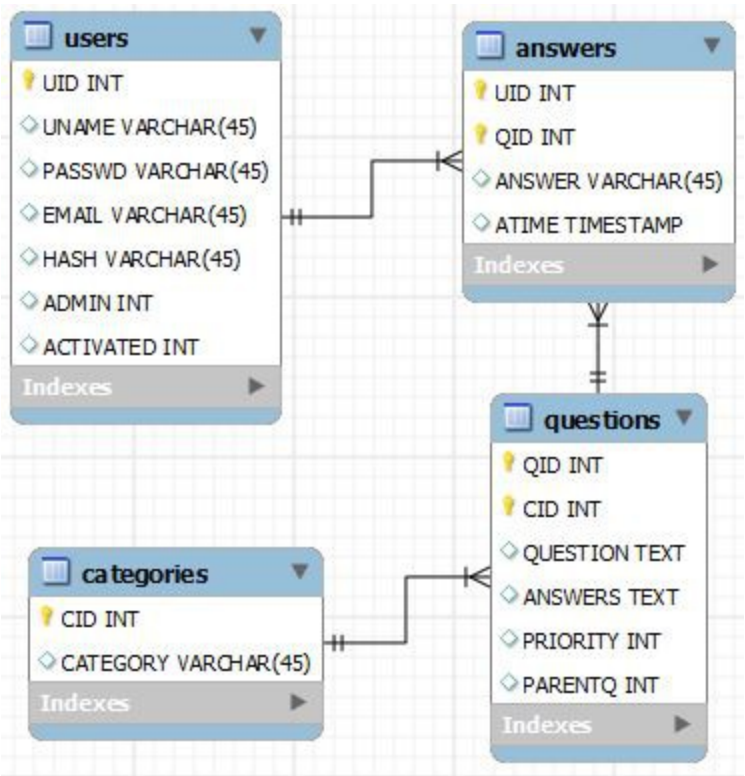
config.php – Konfiguraatiotiedosto. Sisältää tärkeitä sivuston tietoja, esim. tietokantayhteyden asetuksia.

library – Sisältää kaikki sovelluksen luokka- ja funktiokirjastot.

templates – Sisältää ulkoasujen määrittämiseen tarvittavat tiedostot.

3.3 Tietokanta

Seuraavaksi oli vuorossa tietokannan suunnittelu. Suunnittelu aloitettiin sovelluksen tärkeimpien ominaisuuksien perusteella. Koska sovellus perustuu yksilöllisiin käyttäjiin ja sisäänkirjautumiselle, täytyy tietokanta sisältää taulu käyttäjiä varten. Käyttäjätaulun lisäksi tarvitaan taulu kysymyksiä varten. Kysymyksillä on myös kategoriat, joille tarvitaan oma taulunsa. Viimeinen tarvittava taulu on vastaustaulu, johon kaikkien käyttäjien vastaukset tallennetaan.



Kuva 3. Tietokantamalli.

Kuten kuvasta 3 näkyy, sovellus sisältää kaiken kaikkiaan neljä taulua. Kuvasta 3 näkee myös, että UID, QID ja CID ovat pääavaimia (primary key), jotka

yksilöivät tietueet. UID yksilöi käyttäjät, QID yksilöi kysymykset ja CID yksilöi kategoriat. Vastauksille ei kuitenkaan ole omaa erillistä yksilöllistä avainta, vaan se käyttää kahta edellä mainittua avainta yksilöimiseen. Vastaukset yksilöidään siis käyttäjän tunnisteella UID ja kysymyksen tunnisteella QID. Näin voidaan tehdä, koska yhdellä käyttäjällä ei voi olla yhteen kysymykseen kahta eri vastausta. Alla vielä listaus ja selitys tauluista ja niiden sarakkeista.

users – Taulu käyttäjien tietoja varten.

UID (INT) – Automaattisesti kasvava yksilöllinen tunniste käyttäjille.

UNAME (VARCHAR) – Käyttäjänimi.

PASSWD (VARCHAR) – Sha1 salattu salasana.

EMAIL (VARCHAR) – Sähköpostiosoite.

HASH (VARCHAR) – Tilin aktivoimista varten luotu salausavain.

ADMIN (INT) – Määritetään onko käyttäjällä ylläpito-oikeudet sivustolle.

ACTIVATED (INT) – Määritellään onko käyttäjän tili aktivoitu.

questions – Taulu kysymyksiä varten.

QID (INT) – Automaattisesti kasvava yksilöllinen tunniste kysymyksille.

CID (INT) – Kategoriaan viittaava tunniste.

QUESTION (TEXT) – Kysymys.

ANSWERS (TEXT) – Vastaukset.

PRIORITY (INT) – Kysymyksen tärkeyden määritys.

PARENTQ (INT) – Isäntäkysymyksen määritys.

categories – Taulu kategorioita varten.

CID (INT) – Automaattisesti kasvava yksilöllinen tunniste kategorioille.

CATEGORY (VARCHAR) – Kategorian nimen määrittäminen.

answers – Taulu vastauksia varten.

UID (INT) – Käyttäjään viittaava tunniste

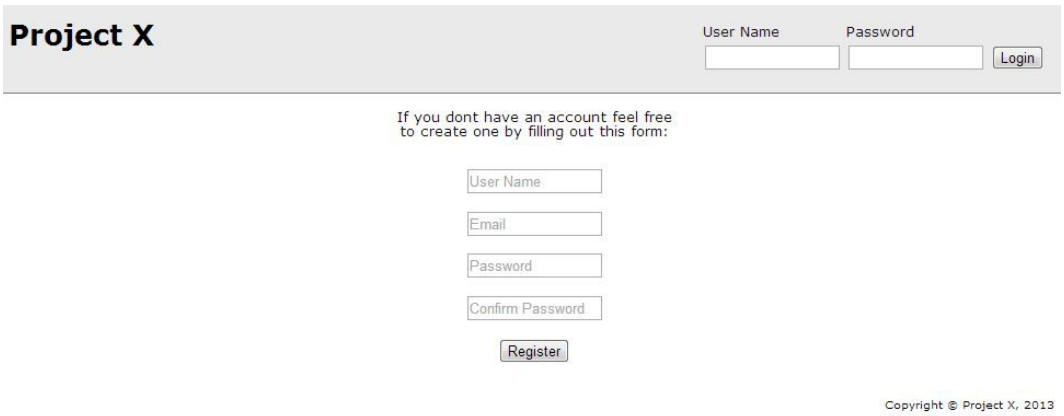
QID (INT) – Kysymykseen viittaava tunniste.

ANSWER (VARCHAR) – Vastaus kysymykseen.

ATIME (TIMESTAMP) – Vastausaika.

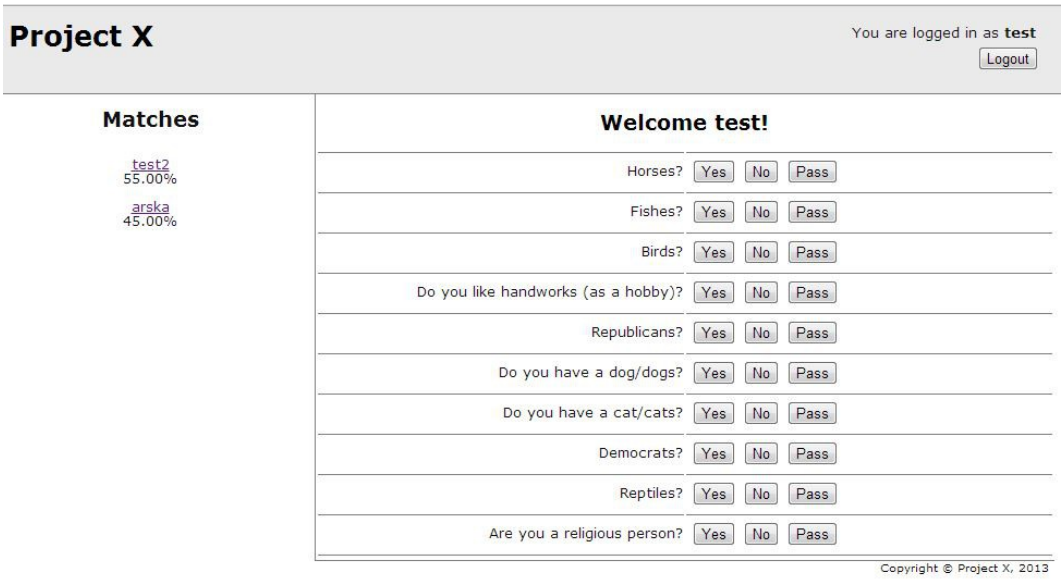
3.4 Ulkoasu

Ulkoasu ei ollut suuressa roolissa tässä työssä, mutta jatkokehityksen kannalta jotain yksinkertaisia ulkoasuun liittyviä asioita täytyi ottaa huomioon. Sivustoa täytyy pystyä selaamaan hieman pienemmän resoluutoin omaavalla päätelaitteella, kuten esim. miniläppärillä tai tabletilla. Tämän huomioon ottaen päädyttiin 980 pixelin vähimmäisleveyteen, joka mahdollistaa sovelluksen saumattoman ulkoasun suurimmille osalle laitteista. Ulkoasu jaottuu kahteen osaan, sisäänkirjautumiseen ja näkymään kun käyttäjä on kirjautuneena sisään. Molemmissa ulkoasuissa kuitenkin käytetään kahta tärkeää osaa, joita ovat yläpalkki ja sisältöosio.



Kuva 4. Ulkoasu sisäänkirjautumiselle.

Kuvasta 4 näkee miltä sisäänkirjautumisnäkymä näyttää. Kuvasta 4 voi myös nähdä miten näkymä on jaoteltu kahteen osaan vaakaviivalla. Yläpalkissa on sivuston nimi/logo ja sisäänkirjautumislomake. Sisältöosio sisältää rekisteröitymislomakkeen.



Kuva 5. Näkymä sisäänkirjautumisen jälkeen.

Kuvasta 5 näkee saman jaottelun kuin ennen sisäänkirjautumista eli yläpalkki, joka sisältää sivuston nimen/logon ja uloskirjautumispainikkeen. Yläpalkki näyttää myös sisäänkirjautuneen näyttäjän nimen. Sisältöosio on tässä näkymässä jaettu kahteen osaan. Vasemmassa palkissa näytetään vastausten perusteella tehdyn vertailun tulokset. Oikeassa palkissa näytetään kysymykset, joihin käyttäjä

voi vastata.

4 TOTEUTUS

Tässä luvussa käydään läpi työn toteutus tietokantojen luonnista toimivaan sovellukseen. Sovellusta rakennettiin ja testattiin palvelimella, joka sisältää LAMP ohjelmistokokonaisuuden. LAMP tulee sanoista Linux, Apache, MySQL ja PHP. Palvelimessa on käyttöjärjestelmä Ubuntu Server LTS 12.04.2, Apache2 HTTP-palvelin, MySQL tietokanta ja PHP5.

4.1 Tietokannan luonti

Sovelluksen toteutus alkoi tietokannan luonnista. Luomisessa käytettiin web-pohjaista MySQL-tietokannan hallintasovellusta nimeltä phpMyAdmin. Tietokantaa varten luotiin käyttäjä nimeltä oppari, jolle luotiin saman niminen tietokanta eli oppari. Kun tietokanta ja käyttäjä on luotu, voidaan tietokantaan lisätä suunnitteluvaiheessa määritellyt taulut. Taulukko 4 sisältää taulujen luontiin käytetyt SQL-lauseet.

Taulukko 4. SQL-lauseet taulujen luontiin.

```
CREATE  TABLE IF NOT EXISTS `oppari`.`users` (
  `UID` INT NOT NULL ,
  `UNAME` VARCHAR(45) NULL ,
  `PASSWD` VARCHAR(45) NULL ,
  `EMAIL` VARCHAR(45) NULL ,
  `HASH` VARCHAR(45) NULL ,
  `ADMIN` INT NULL ,
  `ACTIVATED` INT NULL ,
  PRIMARY KEY (`UID`) )
ENGINE = InnoDB

CREATE  TABLE IF NOT EXISTS `oppari`.`answers` (
  `UID` INT NOT NULL ,
  `QID` INT NOT NULL ,
  `ANSWER` VARCHAR(45) NULL ,
  `ATIME` TIMESTAMP NULL ,
  PRIMARY KEY (`UID`, `QID`) ,
  INDEX `fk_answers_questions_idx` (`QID` ASC) ,
  INDEX `fk_answers_users1_idx` (`UID` ASC) ,
  CONSTRAINT `fk_answers_questions`
    FOREIGN KEY (`QID` )
```

```
    REFERENCES `oppari`.`questions` (`QID` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_answers_users1`
    FOREIGN KEY (`UID` )
    REFERENCES `oppari`.`users` (`UID` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB

CREATE  TABLE IF NOT EXISTS `oppari`.`questions` (
  `QID` INT NOT NULL ,
  `CID` INT NOT NULL ,
  `QUESTION` TEXT NULL ,
  `ANSWERS` TEXT NULL ,
  `PRIORITY` INT NULL ,
  `PARENTQ` INT NULL ,
  PRIMARY KEY (`QID`, `CID`) ,
  INDEX `fk_questions_categories1_idx` (`CID` ASC) ,
  CONSTRAINT `fk_questions_categories1`
    FOREIGN KEY (`CID` )
    REFERENCES `oppari`.`categories` (`CID` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB

CREATE  TABLE IF NOT EXISTS `oppari`.`categories` (
  `CID` INT NOT NULL ,
  `CATEGORY` VARCHAR(45) NULL ,
  PRIMARY KEY (`CID`) )
ENGINE = InnoDB
```

4.2 Ohjelman tekeminen

Kun taustatyö on tehty voi itse ohjelmointityö alkaa. Suunnitelman mukaisesti luotiin tiedostorakenne kuvan 2 mukaisesti. Resources-kansiossa oleva config.php tiedostossa määritellään tietokantayhteyden tiedot, lyhenteitä tiedostopoluista ja virheiden näyttämisestä. Taulukosta 5 selviää millainen config.php tiedosto on.

Taulukko 5. Config.php.

```
<?php
/**
```

```
 * Tämä tiedosto on asetusten määrittämiseen
 */
$config = array(
      "db" => array(
            "db1" => array(
                  "dbname" => "oppari",
                  "dbusername" => "oppari",
                  "dbpassword" => "projectx",
                  "dbhost" => "localhost"
            )
      ),
      "urls" => array(
            "baseUrl" => "http://rrt.mine.nu/~project/"
      ),
      "paths" => array(
            "resources" => "/home/project/resources",
            "images" => array(
                  "content" => $_SERVER["DOCUMENT_ROOT"] .
"/images/content",
                  "layout" => $_SERVER["DOCUMENT_ROOT"] .
"/images/layout"
            )
      ),
      "emails" => array(
            "baseEmail" => "noreply@rrt.mine.nu"
      )
);
/**
 * Luodaan lyhenteitä tiedostopoluista, että niitä on helpompi
käyttää
 */
defined("LIBRARY_PATH")
      or define("LIBRARY_PATH", realpath(dirname(__FILE__) .
'/library'));
defined("TEMPLATES_PATH")
      or define("TEMPLATES_PATH", realpath(dirname(__FILE__) .
'/templates'));
/**
 * Virheiden näyttäminen
 */
ini_set("error_reporting", "true");
error_reporting(E_ALL|E_STRCT);
ini_set('display_errors', '1');
?>
```

Kun asetukset on määritelty alkoi ohjelmointi. Ohjelmointi suoritettiin siten, että yksi ominaisuus kerrallaan valmiiksi ja sen jälkeen siirryttiin uuteen ominaisuuteen. Testausta tapahtui jatkuvasti ohjelmoinnin edetessä. Kun yksi ominaisuus tuli valmiiksi, testattiin kyseistä ominaisuutta hieman perusteellisemmin.

4.2.1 Näkymä

Sovelluksen näkymä toteutettiin siten, että käyttäjän selatessa näytetään kokoajan /public_html/index.php tidostoa, eikä hänen tarvitse hyppiä alikansioihin tai muihin tiedostoihin. Tämä index.php tiedosto siis sisältää kaiken logiikan ja tiedon siitä, millainen näkymä käyttäjälle näytetään. Tämä toteutettiin siten, että index.php kutsuu /resources/library/templateFunctions.php tiedostosta löytyviä funktioita, jotka palauttaa parametrien mukaan halutun näkymän.

Taulukko 6. Esimerkki käytetystä näkymä-funktiosta.

```
    /**
     * Luodaan login näkymä
     *
     * @param string $contentFile sisältötiedosto
     * @param array $variables muuttujalista index.php
tiedostosta
     */
      function renderLayoutLogin($contentFile, $variables =
array()){
          //Haetaan config.php tiedostosta tempates kasion polku
ja sieltä haluttu tiedosto
            $contentFileFullPath = TEMPLATES_PATH . "/" .
$contentFile;
            //Tarkistetaan tuotiinko parametreja index.php:sta
            if (count($variables) > 0) {
                  foreach ($variables as $key => $value) {
                        if (strlen($key) > 0) {
                              ${$key} = $value;
                        }
                  }
            }
        //Lisätään näkymään headerLogin.php eli yläpalkki
            require_once(TEMPLATES_PATH . "/headerLogin.php");
            echo "<div id=\"container\">";
            echo "<div id=\"content\">";
        //Tarkistetaan löytyykö haluttua tiedostoa
            if (file_exists($contentFileFullPath)) {
                //Jos tiedosto löytyi, niin lisätään se näkymään
                  require_once($contentFileFullPath);
            } else {
            //Jos haluttua tiedostoa ei löydy, näytetään
error.php sivu
                  require_once(TEMPLATES_PATH . "/error.php");
            }
            // close content div
            echo "</div>";
            //Lisätään näkymään footer.php
            require_once(TEMPLATES_PATH . "/footer.php");
      }
```

4.2.2 Tietokantayhteys

Ennen sisäänkirjautumista tarvitaan tietysti tietokantayhteys, jolla voidaan muodostaa yhteys tietokantaan tarvittavien tietojen tarkastamista varten. Tietokantayhteyttä varten luotiin luokka, joka sisältää tietokantayhteyden avaamisen ja sulkemisen. Taulukkossa 7 on kyseinen luokka kokonaisuudessaan.

Taulukko 7. Tietokantayhteys luokka.

```
<?php
//Tietokanta luokka
class DBConn {
      //Muuttuja yhteyttä varten
      private $conn;

      /**
     * Tietokantayhteyden avaaminen
     *
     * @param array $db_configs
     */
      function openDBConn($db_configs){
            //Muodostetaan tietokantayhteys annetuilla
asetuksilla
            $this->conn = mysql_connect($db_configs['dbhost'],
$db_configs['dbusername'], $db_configs['dbpassword']);
            //Jos tietokantayhteyttä ei voitu muodostaa
            if (!$this->conn) {
                //Suljetaan yhteys error viestin kera
                  die('Could not connect: ' . mysql_error());
        //Jos tietokantayhteys saatiin muodostettua
            }else{
                  //Valitaan tietokanta
                  $db_selected =
mysql_select_db($db_configs['dbname'], $this->conn);
            }
            //Jos tietokantaa ei voitu valita
            if (!$db_selected) {
                //Suljetaan yhteys error viestin kera
                  die('Can\'t use ' . $dbname . ' : ' .
mysql_error());
            }
      }

      /**
     * Tietokantayhteyden sulkeminen
     */
      function closeDBConn(){
            //Suljetaan tietokantayhteys
            mysql_close($this->conn);
      }
} ?>
```

4.2.3 Sisäänkirjautuminen

Kun näkymän logiikka on tehty, siirrytään sisäänkirjautumisen luomiseen. Sisäänkirjautumiseen liittyvät funktiot luotiin tiedostoon /resources/sessionFunctions.php, jossa tarkistetaan onko käyttäjä kirjautumassa sisään, kirjautunut jo sisään vai kirjautunut ulos. Kyseisessä tiedostossa on funktio istunnon luomiseen ja istunnon tarkistamiseen. Koska sisäänkirjautuminen vaikuttaa näkymään, mikä käyttäjälle näytetään, käyttää index.php näitä tarkistuksia, joiden perusteella index.php kutsuu edellä mainittuja näkymäfunktioita tarkastusten perusteella. Esimerkkinä voidaan esittää tilanne, jossa käyttäjä avaa sivun ensimmäistä kertaa. Ensimmäisenä index.php tarkastaa /resources/sessionFunctions.php -tiedostosta löytyvällä funktiolla onko käyttäjä kirjautuneena sisään. Koska käyttäjä vierailee ensimmäistä kertaa sivustolla, ei hän voi olla kirjautuneena sisään eikä hänellä voi olla voimassaolevaa istuntoa. Tarkistuksen perusteella kutsutaan taulukossa 6 esiintyvää funktiota, jolla näytetään sisäänkirjautumisnäkymä. Taulukossa 8 on index.php:n kutsuma sisäänkirjautumisen tarkastusfunktio, jonka perusteella oikea näkymä valitaan käyttäjälle.

Taulukko 8. Sisäänkirjautumisen tarkistusfunktio.

```
/**
 * Sisäänkirjautumisen tarkistaminen
 *
 * @return int
 */
function loginControl(){
    //Tämänhetkinen aika
      $currenttime = time();
    //Jos istunnossa on 'login' parametri ja sen arvo on 1
      if(isset($_SESSION['login']) && $_SESSION['login']==1){
          //Jos käyttäjä on painanut uloskirjautumista
            if(isset($_POST['logout'])){
                //Tyhjätään istunto
                  session_unset();
            //Tuhotaan istunto
                  session_destroy();
            //Palautetaan tieto, että käyttäjä on kirjautunut
ulos
                  return(0);// YOU ARE NOW LOGGED OUT
            }
```

```
        //Jos istunto on vanhentunut
            else if(($currenttime > $_SESSION['expire']) &&
validSession()){
                //Tyhjätään istunto
                  session_unset();
            //Tuhotaan istunto
                  session_destroy();
            //Palautetaan tieto, että käyttäjän istunto on
vanhentunut
                  return(4);// SESSION EXPIRED
            }
        //Jos sessio on halutunlainen
            else if(validSession()){
                //Palautettaan tieto, että käyttäjän istunto on
voimassa, oikeanlainen ja käyttäjä on kirjautuneena sisään
                  return(1);// SESSION IS VALID AND USER LOGGED
IN
            }
    //Jos käyttäjä on painanut sisäänkirjautumista
      }else if(isset($_POST['username']) &&
strlen($_POST['username'])>0 && isset($_POST['password']) &&
strlen($_POST['password'])>0){
          //Haetaan syötetty käyttäjänimi
          $username = $_POST['username'];
        //Haetaan syötetty salasana
          $password = $_POST['password'];
        //Jos sisäänkirjautumistedot eivät ole oikeat
            if(validateLogin($username,$password)==0){
                //Palautettaan tieto, että käyttäyttäjänimi tai
salasana ovat väärin
                  return(2);// INVALID USERNAME AND PASSWORD
            }
        //Jos käyttäjätunnus ja salasana ovat oikeat
            elseif(validateLogin($username,$password)==1){
                //Palautettaan tieto, että käyttäjätunnus ja
salasana ovat oikein
                  return(3);// VALID USERNAME AND PASSWORD
            }
        //Jos käyttäjätunnus ja salasana ovat oikeat ja
käyttäjällä on admin oikeudet
            else if(validateLogin($username,$password)==3){
                //Palautettaan tieto, että käyttäjätunnus ja
salasana ovat oikeat ja käyttäjällä on admin oikeudet
                  return(7);// VALID USERNAME AND PASSWORD AND
USER IS ADMIN
            }
        //Jos käyttäjätiliä ei ole vielä aktivoitu
            elseif(validateLogin($username,$password)==2){
                //Palautettaan tieto, että käyttäjätiliä ei ole
vielä aktivoitu
                  return(6);// ACCOUNT NOT ACTIVATED
            }
    //Jos käyttäjä ei ole kirjautuneena sisään
      }else{
          //Tyhjätään istunto
            session_unset();
```

```
        //Tuhotaan istunto
            session_destroy();
            //Palautettaan tieto, että käyttäjä ei ole
kirjautuneena sisään
            return(5);// NOT LOGGED IN
      }
}
```

Taulukko 9 sisältää osan index.php sisältämästä tarkastuksesta, joka kutsuu taulukon 8 funktiota.

Taulukko 9. Osa index.php:n sisältämästä tarkastuksesta.

```
//jos käyttä on painanut logout -nappia ohjataan hänet
sisäänkirjautumissivulle viestin kera
      if($logincontrol==0){
          //viesti joka näytetään etusivulla
            $errormsg = "You are now logged out!";
            //luodaan array joka lähetetään viestin
templateFunction.php -tiedostoon, jonka kautta se näytetään
etusivulla
            $variables = array(
                  'errormsg' => $errormsg
            );
        //Kutsutaan renderLayoutLogin -funktio jossa kutsutaan
etusivuksi kirjautumissivu login.php edellä määritetyillä
parametreilla
            renderLayoutLogin("login.php", $variables);
    }
```

4.2.4 Rekisteröinti

Etusivulla oleva rekisteröintilomake, joka näytetään sisäänkirjautumisnäkymässä, läheteään jos käyttäjä täyttää rekisteröintilomakkeen ja painaa rekisteröintipainiketta (Register). Lomake itse tarkistaa, onko tarvittavat kentät täytetty oikein. Kun tarvittavat tiedot on täytetty oikein, tallennetaan käyttäjän tiedot tietokantaan. Tämän yhteydessä tietokannan users-taulun hash-sarakkeeseen luodaan yksilöllinen avain, jonka avulla käyttäjätilin voidaan aktivoida. Tätä hash-avainta ja sähköpostiosoitetta käyttäen muodostetaan linkki, joka lähetetään käyttäjälle. Linkkiä painamalla käyttäjän tili aktivoidaan.

4.3 Kysymysten hakeminen

Kun käyttäjätoiminnot saatiin tehtyä, alkoi kysymysten tuominen käyttäjälle. Kysymysten hakeminen toimii siten, että kun käyttäjä kirjautuu sivustolle, hänelle haetaan 20 kysymystä varastoon. Aina kun käyttäjä vastaa yhteenkin kysymykseen tästä varastosta, lisätään tämä vastaus tietokantaan, poistetaan kyseinen kysymys varastosta ja haetaan uusi kysymys vastatun tilalle. Syöttämistä varten tietokantaan lisättiin testidataa, jonka avulla toiminnon tekeminen voitiin suorittaa. Aluksi tehtiin luokka, joka sisältää kaikki kysymyksiin liittyvät funktiot. Luokkaan luotiin funktio, joka hakee kysymyksiä käyttäjälle tietyillä parametreilla. Parametreina ovat käyttäjän id, montako uutta kysymystä haetaan ja käyttäjällä varastossa olevat kysymykset. Näiden perusteella käyttäjälle haetaan uusia kysymyksiä. Osalla kysymyksistä on asetettuna parentq-sarake, jota käytetään uusien kysymysten hakemista varten. Esimerkkinä tapaus, jossa kysymykselle, jonka id on 50, on asetettu isäntäkysymys parentq, jonka id on 1. Tämä tarkoittaa siis, että kyseisen kysymyksen 50 isäntäkysymyksen id on 1. Jos käyttäjä on vastannut kysymykseen 1 myönteisesti, hänelle voidaan hakea kysymys 50. Jos taas käyttäjä vastaa kysymykseen 1 kielteisesti, häneltä ei kysytä kysymystä 50.

4.3.1 Kysymyksiin vastaaminen

Kun kysymykset on saatu näkymään käyttäjälle, lisätään käyttäjälle mahdollisuus vasta esitettyihin kysymyksiin. Kysymys-luokkaan lisättiin funktio vastausten asettamista varten. Funktiolle syötetään käyttäjän id, kysymyksen id ja vastaus kysyttyyn kysymykseen. Näin funktio voi lisätä vastauksen tietokantaan. Jotta kysymyksiin vastaamisen ei lataa koko sivua aina uudelleen, käytettiin vastauksien lähettämiseen Ajaxia ja jQueryä. Kun käyttäjä vastaa kysymykseen, lähetetään vastaus Ajaxilla ”post” muodossa vastausten käsittelijälle, joka lähettää vastauksen Ajax funktiolle. Saatu tulos, eli uusi lista kysymyksistä, näytetään käyttäjälle. Taulukosta 10 selviää millainen opinnäytetyössä käytetty Ajax-funktio on.

Taulukko 10. Kysymykseen vastaamiseen käytetty Ajax-funktio.

```
var $j = jQuery.noConflict();

        $j(document).ready(function(){
            $j("input[id='buttons']").click(function(){
                var bufferdata = $j("#bufferForm").serialize();
                var buttondata = $j(this).attr("name") + "=" +
$j(this).val();
                var wholedata = bufferdata +"&"+ buttondata;
                $j.ajax({
                        url:"http://rrt.mine.nu/~project/",
                        type:"post",
                        data: wholedata,
                        success:function(data){
                            $j("body").html(data);
                        }
                });
                return false;
           });
```

4.3.2 Vastausten vertaileminen

Käyttäjän vastattua kysymyksiin tarvitaan tietenkin tärkein ominaisuus opinnäytetyössä, eli vastausten vertaileminen käyttäjien välillä. Tämän toteuttamiseen kului todella paljon aikaa sen monimutkaisuuden vuoksi. Vertailun toteutusta ei haluttu toteuttaa PHP:lla, vaan suoraan SQL-kyselyllä, joka tekee vertailusta huomattavasti nopeampaa. Tämä toteutustapa vaikeutti myös vertailun toteuttamista. SQL-kyselyn luominen onnistui kuitenkin lopulta. Taulukosta 11 näkee kyseisen kyselyn kokonaisuudessaan. Kyselylle syötetään halutun käyttäjän id, jonka vastauksia halutaan verrata muiden käyttäjien samoihin vastauksiin. Nämä vastaukset pisteytetään kysymyksen prioriteetin perusteella ja verrataan muiden käyttäjien pisteisiin. Pisteistä lasketaan paljonko ne prosentuaalisesti eroavat toisistaan.

Taulukko 11. SQL-kysely vastausten vertailemiseen.

```
SELECT (
((SUM(points)*100))
/
(SELECT SUM(points) FROM
(
```

```
SELECT (COUNT(*) * 4) AS points
FROM answers AS a
WHERE a.UID='$uid' AND a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=1)
UNION ALL
SELECT (COUNT(*) * 3) AS points
FROM answers AS a
WHERE a.UID='$uid' AND a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=2)
UNION ALL
SELECT (COUNT(*) * 2) AS points
FROM answers AS a
WHERE a.UID='$uid' AND a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=3)
UNION ALL
SELECT (COUNT(*) * 1) AS points
FROM answers AS a
WHERE a.UID='$uid' AND a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=4)
UNION ALL
SELECT (COUNT(*) * 0.1) AS points
FROM answers AS a
WHERE a.UID='$uid' AND a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=5)
) x)
)
AS percent
FROM (SELECT (COUNT(*) * 4) AS points
FROM answers AS b
WHERE b.UID='$uid' AND
b.QID IN (SELECT a.QID
     FROM answers AS a
     WHERE a.UID='$uid2' AND a.ANSWER=b.ANSWER AND
a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=1))
UNION ALL
SELECT (COUNT(*) * 3) AS points
FROM answers AS b
WHERE b.UID='$uid' AND
b.QID IN (SELECT a.QID
```

```
     FROM answers AS a
     WHERE a.UID='$uid2' AND a.ANSWER=b.ANSWER AND
a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=2))
UNION ALL
SELECT (COUNT(*) * 2) AS points
FROM answers AS b
WHERE b.UID='$uid' AND
b.QID IN (SELECT a.QID
     FROM answers AS a
     WHERE a.UID='$uid2' AND a.ANSWER=b.ANSWER AND
a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=3))
UNION ALL
SELECT (COUNT(*) * 1) AS points
FROM answers AS b
WHERE b.UID='$uid' AND
b.QID IN (SELECT a.QID
     FROM answers AS a
     WHERE a.UID='$uid2' AND a.ANSWER=b.ANSWER AND
a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=4))
UNION ALL
SELECT (COUNT(*) * 0.1) AS points
FROM answers AS b
WHERE b.UID='$uid' AND
b.QID IN (SELECT a.QID
     FROM answers AS a
     WHERE a.UID='$uid2' AND a.ANSWER=b.ANSWER AND
a.QID IN
     (SELECT q.QID FROM questions AS q WHERE
PRIORITY=5))
) x;
```

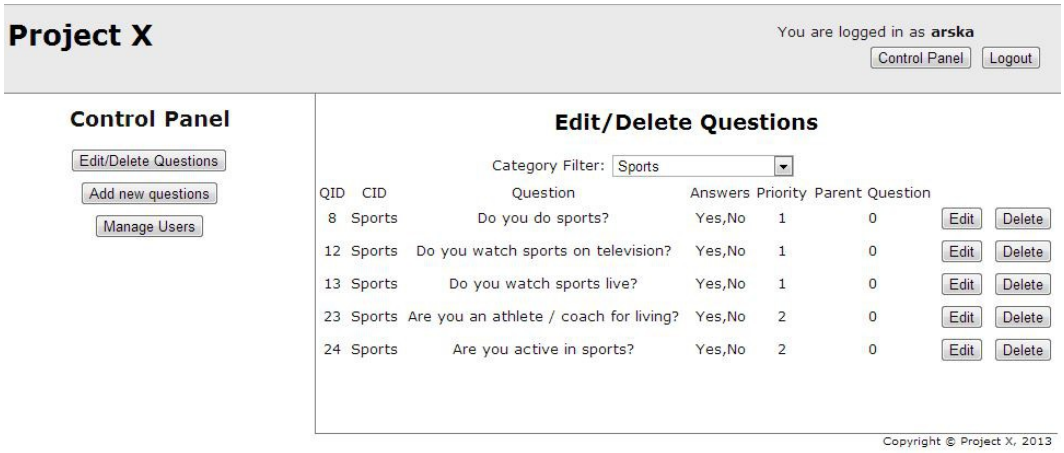
Vertailu suoritetaan vasta kun käyttäjä on vastannut 20:en ensimmäiseen pakolliseen kysymykseen. Vertailun tuloksena näytetään 5 parhaiten vertailussa pärjännyttä käyttäjää. Käyttäjien vertailu näytetään käyttäjälle vasemmassa palkissa kuten kuvasta 5 voi todeta.

4.4 Yhteydenotto käyttäjien välillä

Yhteydenottoa varten tehtiin linkki nimiin, jotka saatiin vertailun tuloksena. Linkki tuo käyttäjälle kysymysten tilalle kentän, johon hän voi syöttää haluamansa viestin käyttäjälle, jonka käyttäjänimeä hän painoi. Viestin kirjoitettuaan käyttäjä voi painaa lähetä viesti (Send message) painiketta, joka lähettää viestin sähköpostina valitulle käyttäjälle.

4.4.1 Admin työkalut

Helpottaakseen käyttäjien ja kysymysten hallintaa lisättiin admin työkaluja. Näillä työkaluilla voi sekä lisätä, muokata ja poistaa kysymyksiä että muokata ja poistaa käyttäjiä. Kaikkiin näihin ominaisuuksiin käytettiin Ajaxia, että muokkaaminen, lisääminen ja poistaminen tapahtuvat dynaamisesti. Admin näkymässä lisättiin yläpalkkiin painike, joka avaa vasempaan valikkoon listan työkaluista ja työkalupainiketta painamalla avautuu kyseinen työkalu oikeaan valikkoon. Kuvasta 6 voi nähdä adminin kysymysten listaus näkymän.



Kuva 6. Admin näkymä kysymysten listauksella.

4.5 Lisäkehitys

Kehitysideoita, joita työn suorituksen ohessa tuli esille, tuli muutamia vastaan. Yksi kehitysideoista oli yhteydenpidon muuttaminen chat -tyyliseksi sähköpostien lähettämisen sijaan. Toinen kehitysidea oli sovelluksen ulkonäön kehittäminen,

koska suoritusajan vuoksi työssä ei kiinnitetty huomiota sovelluksen ulkoasuun.

5 YHTEENVETO

Työn suorituksen alussa määritetyt pakolliset toiminnot sain toteutettua ja sen lisäksi vielä admin työkalut, jotka helpottivat sovelluksen ylläpitäjää. Ottaen huomioon sen, että en ole opiskellut oppilaitoksessa lainkaan PHP:ta ja sen johdosta opiskelin sen omatoimisesti, työn suoritus onnistui mielestäni erinomaisesti. Ajax ja jQuery olivat ennen tätä työtä tuntemattomia minulle, mutta niidenkin perustoiminnot tulivat todella tutuksi työn ohessa. PHP:sta opin todella paljon ja yleisesti web-ohjelmoinnista.

Ongelmia syntyi päivittäin joskus ratketen lähes välittömästi ja joskus taas joutui iltaisin sängyssä pohtia pitkään kun ei saanut ongelmaa ratkaistua ja pois mielestä. Jokapäiväiseen PHP:n ongelmanratkaisuun suuren avun toivat sivustot Stack Overflow ja PHP.NET. Näiden sivustojen avulla työn suoritus onnistui sujuvasti. Kuitenkin kaikki ongelmat ratkesivat ja varsinkin vastausten vertailuun käytetty SQL-kyselyn ratkaiseminen toi mahtavan onnistumisen tunteen. Tämän sovelluksen innoittamana päädyinkin opettelemaan lisää web-ohjelmointia ja sen johdosta halusinkin ja pääsinkin ohjelmoimaan web-sovelluksia.

Työstä kokonaisuutena antaisin hyvän arvosanan edellä mainitut asiat huomioon ottaen. Sovellus toimii, se on selkeä ja sitä on helppo käyttää. Sitä on myös helppo kehittää jatkossa. Ajankäyttö olisi voinut olla parempaa kirjallista työtä tehdessä, mutta siitäkin selvisin mielestäni todella hyvin.

LÄHTEET

PHP.NET, 2013. Viitattu 28.4.2013.

<http://php.net/>

Stack Overflow, 2013. Viitattu 28.4.2013 .

<http://stackoverflow.com/>

W3schools, 1999-2013. Introduction to SQL. Viitattu 28.4.2013.

<http://www.w3schools.com/>

Nettuts+, 2013. Organize Your Next PHP Project the Right Way. Viitattu 28.4.2013.

<http://net.tutsplus.com/>

Zandstra, Matt 2005. PHP Trainer Kit. 3 Painos. Helsinki. Erita Prima Oy.